

SDEV 1201

Database Fundamentals

LESSON 4

Let's review

Simple SELECT

```
SELECT  
    FirstName  
, LastName  
, City  
FROM Student;
```

Derived fields need a name!

There is another difference with the query that returns the names separately than the one that returned them as one column. The column in the second query does not have a name. This is a problem. We should give all columns a name. Calculated columns do not have a name because they are calculated from many different columns and values. So, we must give the column a name. In the example, the “AS” keyword is optional.

```
Select FirstName || ' ' || LastName AS "Student Name", City  
From Student;
```

In this course you are mandated to ALWAYS have column names.

Where

So far, we have only selected data that includes ALL the records on the table. While this is not uncommon, it is likely that you will not only want to select the columns you want, but also only the records you want. The WHERE clause allows you to limit the records that are returned based on criteria you provide.

```
Select StudentID, FirstName, LastName, Gender, StreetAddress, City,  
Province, PostalCode, Birthdate, BalanceOwing  
From Student  
Where StudentID = '198933540';
```

This query would return all the student information for the student with StudentID 198933540. The “Where” clause can contain many different expressions to identify the records the query should return.

Like

The Like operator is used when you use wildcards in your search or perform pattern matching. Wildcards are characters that when used with the Like operator allow the wildcards to represent a character or multiple characters. The Like operator is case sensitive in PostgreSQL.

For example, if we wanted to return all the student's names whose names started with the letter 'D' we would use the % wildcard character. The % wildcard means any character and any number of characters.

```
Select FirstName || ' ' || LastName "Student Name"  
From Student  
Where FirstName Like 'D%';
```

It is important to use the like operator when using wildcards. The % would be interpreted as a literal '%' if we used = instead of Like.

Like

The `_` (underscore) is a single character wildcard. This would be used when you are looking for any one character in your search criteria. If we wanted the students' names whose first name was 3 characters long and started with 'Ja' we could use this query.

```
Select FirstName || ' ' || LastName "Student Name"  
From Student  
Where FirstName Like 'Ja%';
```

If we had used a `%` instead of `_` we would also return students whose First Name was longer than 3 characters.

Basic Select Exercise 1

Today's Objective

By the end of this section, you will learn:

- ❑ **Aggregate** functions.
- ❑ **Entity Relationship Diagram Symbols & Terminology Quiz**

Full documentation is on the “2C Basic Select Statement” document located in the Brightspace site in the “Week 4 – Select Statements” section.

Aggregate Functions

Aggregate functions calculate a single value using the data from many records. If you have used function in Excel such as SUM, AVERAGE, COUNT, etc.... then you have used aggregate functions.

aggregate_function_name ([ALL | DISTINCT] expression)

- *AVG(ColumnName)* returns the average of numeric values. **NULL** is ignored.
- *SUM(ColumnName)* returns the sum of a column containing numeric values.
- *MIN(ColumnName)* and *MAX(ColumnName)* returns the minimum & maximum values from a column of numeric, date, or character values.
- *COUNT(*)* returns the number of records that match the **WHERE** criteria.
- *COUNT(ColumnName)* returns the number of non-null values in a given column for records that match the **WHERE** criteria.

Aggregate Functions

AVG

The AVG function returns the average of a column of values. It can only be used on numeric columns and records containing null are ignored and not included in the average calculation.

```
Select AVG(Mark) "Average Mark"  
From Registration  
Where CourseID = 'DMIT2015';
```

Returns the average mark from the registration table for Course DMIT1525. Note that aggregate columns are not columns in the database and therefore do not have a name. Remember, all columns must be given a name if they do not have one.

Aggregate Functions

SUM

The SUM function returns the sum of a column of values. It can only be used on numeric columns and records containing null are ignored and not included in the sum calculation.

```
Select SUM(Amount) "Payment Amount"  
From Payment  
Where StudentID = '200495500';
```

Returns the Sum of all the payments for StudentID 200495500.

Aggregate Functions

MIN

The MIN function returns the minimum value from a column of values. It can be used with columns that have a numeric, date or character datatype. Records with a null value in that column are ignored.

```
Select MIN(Mark) "Lowest Mark"  
From Registration  
Where CourseID = 'DMIT2015';
```

Returns the lowest Mark in DMIT2015.

Aggregate Functions

MAX

The MAX function returns the maximum value from a column of values.

Can be used with columns that have a numeric, date or character datatype. Records with a null value are ignored.

```
Select Max(Mark) 'DMIT152 Highest Mark'  
From Registration  
Where CourseID = 'DMIT152'
```

Returns the highest Mark in DMIT152.

Aggregate Functions

COUNT

The COUNT function returns the number of non-null values from a column of values OR returns the number of records that match the “where” criteria.

```
Select COUNT (DateReleased) "Number of Staff Released"  
From Staff;
```

Returns the number of staff that have been released (number of staff that have a value in the DateReleased column).

```
Select COUNT (*) "Number of Staff Released"  
From Staff;
```

Returns the number of records returned by this query (number of staff RECORDS in the staff table).

Aggregate Functions

NOTE: The * operator is an overloaded operator. This means that it has more than one meaning depending on how it is used. In a column list in a query it means all columns (which should be avoided unless necessary). In the count function it means 'records'.

Class Activity

Please do the “Basic Select Exercise 2” found in Brightspace in the “Week 4 – SELECT statements”.

- As with the last exercise, the examples and practice are based on IQSchool database.

The script to create the tables is located on the Brightspace site in “Week 4 – Select Statements” and named “2C IQSchools JH Create Tables and Load Data Script PostgreSQL”.